

Homework 8 – Due Friday, April 13, 2007

Reminder

- Collaboration is permitted, but you must write the solutions by yourself without assistance, and be ready to explain them orally to a member of the course staff if asked. You must also identify your collaborators. Getting solutions from outside sources such as the Web or students not enrolled in the class is strictly forbidden.
- To facilitate grading, please write down your solution to each problem on a separate sheet of paper. Make sure to include all identifying information and your collaborators on each sheet. Your solutions to different problems will be graded separately, possibly by different people, and returned to you independently of each other.

Exercises These should not be handed in, but the material they cover may appear on exams: non-starred exercises in Chapters 22.1, 22.2, 22.3, 22.4, 24.2, 24.3.

Problems to be handed in

1. (**Graph representations**) Questions in this problem refer to the adjacency list and adjacency matrix representations defined on pages 528–529 of CLRS. Read over those definitions before answering the questions.

You are given a directed graph $G = (V, E)$ with $|V| = n$ vertices and $|E| = m$ edges. Let G^R be the graph obtained by reversing the directions of all the edges in G . For each of the following, give an efficient algorithm and analyze its running time.

If G is given in the adjacency list representation,

- (a) compute the out-degree of each vertex;
- (b) compute the in-degree of each vertex;
- (c) compute the adjacency list representation of G^R .

If G is given in the adjacency matrix representation,

- (d) compute the adjacency matrix of G^R .

2. Give two algorithms to detect whether a given undirected graph has a cycle. If the graph contains a cycle, your algorithms should output one (not all of them, just one). Base the first algorithm on BFS and the second, on DFS. The running time of your algorithms should be the same as the running times of BFS and DFS. Explain why your algorithms are correct and run in the required time.
3. Some friends of yours are working on techniques for coordinating groups of mobile robots. Each robot has a radio transmitter that is used to communicate with a base station, and your friends find that if the robots get too close to one another, then there are problems with interference

among the transmitters. Your friends need to plan the motion of the robots in such a way that each robot gets to its intended destination, but in the process the robots don't come close enough together to cause interference problems.

We can model this problem abstractly as follows. The floor plan of a building is represented by an undirected graph $G = (V, E)$, and there are two robots initially located at nodes a and b in the graph. The robot at node a wants to travel to node c , and the robot at node b wants to travel to node d . This is accomplished by means of a *schedule*: at each time step, the schedule specifies that one of the robots moves across a single edge, from one node to a neighboring node; at the end of the schedule, the robot from node a should be sitting on c and the robot from node b should be sitting on d . A schedule is *interference-free* if there is no point at which the two robots occupy nodes that are at distance $\leq r$ from one another in the graph, for a given parameter r . Assume that the starting nodes a and b are at a distance of greater than r , and so are the ending nodes c and d .

In this problem you'll design an algorithm that finds an interference-free schedule if one exists.

- (a) A *configuration* is a pair of nodes in G , where the two robots are located. Your goal is to get from a given starting configuration (a, b) to a given ending configuration (c, d) , by making "legal" moves between configurations (one robot can change its location to a neighboring node), and choosing only "legal" configurations (the robots must always be at distance at least $r + 1$ from one another).

Define the following graph H : the nodes are all possible configurations of the robots; that is, the set of nodes in H consists of all possible pairs of nodes in G . We join two nodes in H by an edge if they represent configurations that could be consecutive in a schedule: that is, (u, v) is connected to (u', v') if either $u = u'$ and there is an edge in G from v to v' , or if $v = v'$ and there is an edge in G from u to u' .

- i. How many vertices does H have?
 - ii. How many edges does H have?
- (b) Note that paths in H from (a, b) to (c, d) correspond to schedules for the robots. However, the resulting schedules might not be *interference-free*. We'll form a graph H' by deleting certain nodes from H , so that any path in H' is an interference-free schedule.
- i. Which nodes should we delete from H to obtain H' ?
 - ii. Once you have H' , how can you obtain an interference-free schedule? State the running time of the algorithm in terms of $n_{H'}$ and $m_{H'}$, the number of vertices and edges in H' .
- (c) Give an algorithm for computing the adjacency list representation of H' from the adjacency list representation of G . What is the running time of your algorithm?
- (d) State the running time (in terms of n and m) of the algorithm for the original problem resulting from combining parts (c) and (b[ii]).

4. Do exercise 24.3-5 from CLRS. Answer the following:

- (a) How many vertices does G' have?
- (b) Show that the order in which vertices in V are dequeued when BFS is run on G' is the same as the order in which vertices of V are extracted from the priority queue when Dijkstra's algorithm is run on G .